



1.5 — Introduction to iostream: cout, cin, and endl

👤 ALEX¹ ⌚ APRIL 10, 2024

In this lesson, we'll talk more about `std::cout`, which we used in our *Hello world!* program to output the text *Hello world!* to the console. We'll also explore how to get input from the user, which we will use to make our programs more interactive.

The input/output library

The **input/output library** (io library) is part of the C++ standard library that deals with basic input and output. We'll use the functionality in this library to get input from the keyboard and output data to the console. The *io* part of *iostream* stands for *input/output*.

To use the functionality defined within the *iostream* library, we need to include the *iostream* header at the top of any code file that uses the content defined in *iostream*, like so:

```
1 | #include <iostream>
2 |
3 | // rest of code that uses iostream functionality here
```

std::cout

The *iostream* library contains a few predefined variables for us to use. One of the most useful is **std::cout**, which allows us to send data to the console to be printed as text. *cout* stands for "character output".

As a reminder, here's our *Hello world* program:

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     std::cout << "Hello world!"; // print Hello world! to console
6 |
7 |     return 0;
8 | }
```

In this program, we have included *iostream* so that we have access to *std::cout*. Inside our *main* function, we use *std::cout*, along with the **insertion operator** (`<<`), to send the text *Hello world!* to the console to be printed.

std::cout can not only print text, it can also print numbers:

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     std::cout << 4; // print 4 to console
6 |
7 |     return 0;
8 | }
```

This produces the result:

4

It can also be used to print the value of variables:

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     int x{ 5 }; // define integer variable x, initialized with value 5
6 |     std::cout << x; // print value of x (5) to console
7 |     return 0;
8 | }
```

This produces the result:

5

To print more than one thing on the same line, the insertion operator (`<<`) can be used multiple times in a single statement to concatenate (link together) multiple pieces of output. For example:

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     std::cout << "Hello" << " world!";
6 |     return 0;
7 | }
```

This program prints:

Hello world!

Here's another example where we print both text and the value of a variable in the same statement:

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     int x{ 5 };
6 |     std::cout << "x is equal to: " << x;
7 |     return 0;
8 | }
```

This program prints:

```
x is equal to: 5
```

Related content

We discuss what the `std::` prefix actually does in lesson [2.9 -- Naming collisions and an introduction to namespaces](https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/) (<https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/>)².

std::endl

What would you expect this program to print?

```
1 | #include <iostream> // for std::cout
2 |
3 | int main()
4 | {
5 |     std::cout << "Hi!";
6 |     std::cout << "My name is Alex.";
7 |     return 0;
8 | }
```

You might be surprised at the result:

```
Hi!My name is Alex.
```

Separate output statements don't result in separate lines of output on the console.

If we want to print separate lines of output to the console, we need to tell the console when to move the cursor to the next line.

One way to do that is to use `std::endl`. When output with `std::cout`, `std::endl` prints a newline character to the console (causing the cursor to go to the start of the next line). In this context, `endl` stands for "end line".

For example:

```
1 | #include <iostream> // for std::cout and std::endl
2 |
3 | int main()
4 | {
5 |     std::cout << "Hi!" << std::endl; // std::endl will cause the cursor to
    move to the next line of the console
6 |     std::cout << "My name is Alex." << std::endl;
7 |
8 |     return 0;
9 | }
```

This prints:

```
Hi!
My name is Alex.
```

Tip

In the above program, the second `std::endl` isn't technically necessary, since the program ends immediately afterward. However, it serves a few useful purposes.

First, it helps indicate that the line of output is a “complete thought” (as opposed to partial output that is completed somewhere later in the code). In this sense, it functions similarly to using a period in standard English.

Second, it positions the cursor on the next line, so that if we later add additional lines of output (e.g. have the program say “bye!”), those lines will appear where we expect (rather than appended to the prior line of output).

Third, after running an executable from the command line, some operating systems do not output a new line before showing the command prompt again. If our program does not end with the cursor on a new line, the command prompt may appear appended to the prior line of output, rather than at the start of a new line as the user would expect.

Best practice

Output a newline whenever a line of output is complete.

std::cout is buffered

Consider a rollercoaster ride at your favorite amusement park. Passengers show up (at some variable rate) and get in line. Periodically, a train arrives and boards passengers (up to the maximum capacity of the train). When the train is full, or when enough time has passed, the train departs with a batch of passengers, and the ride commences. Any passengers unable to board the current train wait for the next one.

This analogy is similar to how output sent to `std::cout` is typically processed in C++. Statements in our program request that output be sent to the console. However, that output is typically not

sent to the console immediately. Instead, the requested output “gets in line”, and is stored in a region of memory set aside to collect such requests (called a **buffer**). Periodically, the buffer is **flushed**, meaning all of the data collected in the buffer is transferred to its destination (in this case, the console).

Author’s note

To use another analogy, flushing a buffer is kind of like flushing a toilet. All of your collected “output” is transferred to ... wherever it goes next. Eew.

This also means that if your program crashes, aborts, or is paused (e.g. for debugging purposes) before the buffer is flushed, any output still waiting in the buffer will not be displayed.

Key insight

The opposite of buffered output is unbuffered output. With unbuffered output, each individual output request is sent directly to the output device.

Writing data to a buffer is typically fast, whereas transferring a batch of data to an output device is comparatively slow. Buffering can significantly increase performance by minimizing the number of slow transfers that need to be performed when there are multiple output requests.

std::endl vs ‘\n’

Using `std::endl` can be a bit inefficient, as it actually does two jobs: it moves the cursor to the next line of the console, and it flushes the buffer. When writing text to the console, we typically don’t need to flush the buffer at the end of each line. It’s more efficient to let the system flush itself periodically (which it has been designed to do efficiently).

Because of this, use of the ‘\n’ character is typically preferred instead. The ‘\n’ character moves the cursor to the next line of the console, but doesn’t request a flush, so it will often perform better. The ‘\n’ character is also more concise since it’s both shorter and can be embedded into existing text.

Here’s an example that uses ‘\n’ in two different ways:

```
1  #include <iostream> // for std::cout
2
3  int main()
4  {
5      int x{ 5 };
6      std::cout << "x is equal to: " << x << '\n'; // Using '\n' standalone
7      std::cout << "And that's all, folks!\n"; // Using '\n' embedded into a
double-quoted piece of text (note: no single quotes when used this way)
8      return 0;
9  }
```

This prints:

```
x is equal to: 5
And that's all, folks!
```

When `\n` is used by itself to move the cursor to the next line of the console, it should be single quoted. When embedded into text that is already double-quoted, additional quotes aren't needed.

We'll cover what `\n` is in more detail when we get to the lesson on chars ([4.11 -- Chars](https://www.learncpp.com/cpp-tutorial/chars/) (<https://www.learncpp.com/cpp-tutorial/chars/>)³).

Best practice

Prefer `'\n'` over `std::endl` when outputting text to the console.

Warning

`'\n'` uses a backslash (as do all special characters in C++), not a forward slash.

Using a forward slash (e.g. `'/'n'`) or including other characters inside the single quotes (e.g. `' \n'` or `'.\n'`) will result in unexpected behavior. For example, `std::cout << '/'n';` will often print as `12142`, which probably isn't what you were expecting.

std::cin

`std::cin` is another predefined variable that is defined in the `iostream` library. Whereas `std::cout` prints data to the console using the insertion operator (`<<`), `std::cin` (which stands for “character input”) reads input from keyboard using the **extraction operator** (`>>`). The input must be stored in a variable to be used.

```
1  #include <iostream> // for std::cout and std::cin
2
3  int main()
4  {
5      std::cout << "Enter a number: "; // ask user for a number
6
7      int x{}; // define variable x to hold user input (and value-
8      initialize it)
9      std::cin >> x; // get number from keyboard and store it in variable x
10
11     std::cout << "You entered " << x << '\n';
12     return 0;
13 }
```

Try compiling this program and running it for yourself. When you run the program, line 5 will print “Enter a number: “. When the code gets to line 8, your program will wait for you to enter input. Once you enter a number (and press enter), the number you enter will be assigned to variable `x`. Finally, on line 10, the program will print “You entered ” followed by the number you just entered.

For example (I entered 4):

```
Enter a number: 4
You entered 4
```

This is an easy way to get keyboard input from the user, and we will use it in many of our examples going forward. Note that you don't need to use `'\n'` when accepting input, as the user will need to press the *enter* key to have their input accepted, and this will move the cursor to the next line of the console.

If your screen closes immediately after entering a number, please see lesson [0.8 -- A few common C++ problems](https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/) (<https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/>)⁴ for a solution.

Just like it is possible to output more than one bit of text in a single line, it is also possible to input more than one value on a single line:

```
1  #include <iostream> // for std::cout and std::cin
2
3  int main()
4  {
5      std::cout << "Enter two numbers separated by a space: ";
6
7      int x{}; // define variable x to hold user input (and value-initialize
8      it)
9      int y{}; // define variable y to hold user input (and value-initialize
10     it)
11     std::cin >> x >> y; // get two numbers and store in variable x and y
12     respectively
13
14     std::cout << "You entered " << x << " and " << y << '\n';
15
16     return 0;
17 }
```

This produces the output:

```
Enter two numbers separated by a space: 5 6
You entered 5 and 6
```

Best practice

There's some debate over whether it's necessary to initialize a variable immediately before you give it a user provided value via another source (e.g. `std::cin`), since the user-provided value will just overwrite the initialization value. In line with our previous recommendation that variables should always be initialized, best practice is to initialize the variable first.

We'll discuss how `std::cin` handles invalid input in a future lesson ([9.5 -- std::cin and handling invalid input](https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/) (<https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/>)⁵). For now, it's enough to know that `std::cin` will extract as much as it can, and any input characters that could not be extracted are left for a later extraction attempt.

For advanced readers

The C++ I/O library does not provide a way to accept keyboard input without the user having to press *enter*. If this is something you desire, you'll have to use a third party library. For console applications, we'd recommend [pdcurses](https://pdcurses.org/) (<https://pdcurses.org/>)⁶, [FTXUI](https://github.com/ArthurSonzogni/FTXUI) (<https://github.com/ArthurSonzogni/FTXUI>)⁷, [cpp-terminal](https://github.com/jupyter-xeus/cpp-terminal) (<https://github.com/jupyter-xeus/cpp-terminal>)⁸, or [notcurses](https://github.com/dankamongmen/notcurses) (<https://github.com/dankamongmen/notcurses>)⁹. Many graphical user interface libraries have their own functions to do this kind of thing.

Summary

New programmers often mix up `std::cin`, `std::cout`, the insertion operator (`<<`) and the extraction operator (`>>`). Here's an easy way to remember:

- `std::cin` and `std::cout` always go on the left-hand side of the statement.
- `std::cout` is used to output a value (cout = character output)
- `std::cin` is used to get an input value (cin = character input)
- `<<` is used with `std::cout`, and shows the direction that data is moving (if `std::cout` represents the console, the output data is moving from the variable to the console).
`std::cout << 4` moves the value of 4 to the console
- `>>` is used with `std::cin`, and shows the direction that data is moving (if `std::cin` represents the keyboard, the input data is moving from the keyboard to the variable).
`std::cin >> x` moves the value the user entered from the keyboard into x

We'll talk more about operators in lesson [1.9 -- Introduction to literals and operators](https://www.learncpp.com/cpp-tutorial/introduction-to-literals-and-operators/) (<https://www.learncpp.com/cpp-tutorial/introduction-to-literals-and-operators/>)¹⁰.

Quiz time

Question #1

Consider the following program that we used above:

```
1 | #include <iostream> // for std::cout and std::cin
2 |
3 | int main()
4 | {
5 |     std::cout << "Enter a number: "; // ask user for a number
6 |     int x{}; // define variable x to hold user input
7 |     std::cin >> x; // get number from keyboard and store it in variable x
8 |     std::cout << "You entered " << x << '\n';
9 |     return 0;
10| }
```

The program expects you to enter an integer value, as the variable x that the user input will be put into is an integer variable.

Run this program multiple times and describe what happens when you enter the following types of input instead:

a) A letter, such as *h*

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

b) A number with a fractional part (e.g. 3.2). Try numbers with fractional parts less than 0.5 and greater than 0.5 (e.g. 3.2 and 3.7).

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

c) A small negative integer, such as -3

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

d) A word, such as *Hello*

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

e) A really big number (at least 3 billion)

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

f) A small number followed by some letters, such as *123abc*

[Show Solution](#) ([javascript:void\(0\)](#))¹¹

Related content

We discuss how `std::cin` and `operator>>` handle invalid input in future lesson [9.5 -- std::cin and handling invalid input](#) (<https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/>)⁵.



[Next lesson](#)

1.6 [Uninitialized variables and undefined behavior](#)

13



[Back to table of contents](#)

14



[Previous lesson](#)

1.4 [Variable assignment and initialization](#)

15

16

[B](#) [U](#) [URL](#) [INLINE CODE](#) [C++ CODE BLOCK](#) [HELP!](#)

Leave a comment...

Name*

Email*



Notify me about replies:



POST COMMENT

Find a mistake? Leave a comment above!?

 Avatars from <https://gravatar.com/>¹⁹ are connected to your provided email address.

752 COMMENTS

Newest ▼

**qwerty**

April 9, 2024 3:38 am

for this quiz b) A number with a fractional component. Try numbers with fractional components less than 0.5 and greater than 0.5 (e.g. 3.2 and 3.7).

The fractional component is dropped.

my output was 0 is that correct i don't fully understand what fractional component is dropped means ?

0

Reply

**Alex** Author Reply to [qwerty](#)²⁰ April 10, 2024 4:56 pm

Dropped means it is discarded. For example, if you enter 3.2, the 0.2 part of the value is discarded, leaving you with the value 3.

0

Reply

**rookie**

April 7, 2024 8:08 pm

```

1 | #include <iostream> // for std::cout and std::cin
2 |
3 | int main()
4 | {
5 |     std::cout << "Enter a number: "; // ask user for a number
6 |
7 |     int x{1}; // define variable x to hold user input (and value-
8 | initialize it)
9 |     std::cin >> x; // get number from keyboard and store it in variable x
10 |
11 |     std::cout << "You entered " << x << '\n';
12 |     return 0;
13 | }

```

I would like to ask why when I input letters into this program, the output is 0. I have initialized x to 1. Since the input failed, why does x output 0?

👍 0 ➡ Reply



Alex Author

🗨 Reply to [rookie](#)²¹ ⌚ April 8, 2024 9:18 pm

A failed extraction zeros the variable being extracted to. That way, if the variable wasn't previously initialized, it will still have a known value after the failed extraction.

This is covered in lesson <https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/>

👍 0 ➡ Reply



Mason

⌚ March 29, 2024 2:17 pm

```

1 | #include <iostream>
2 |
3 | int main()
4 | {
5 |     //try to output two newline characters
6 |     std::cout << '\n\n'; //returns 2570
7 |
8 |     std::cout << '\n';
9 |
10 |    //try to output three newline characters
11 |    std::cout << '\n\n\n'; // returns 657930
12 |
13 |    return 0;
14 | }

```

I know how to fix the program but I was wondering why I'm not able to do multiple newline characters in a single quote. Thank you!

👍 0 ➡ Reply

**Alex** AuthorReply to [Mason](#)²² ⌚ March 30, 2024 1:04 pm

Single quotes only support a single character (`'\n'` is treated as a single character). If you want to support multiple characters, use double quotes.

👍 0 ➡ Reply

**Mason**Reply to [Alex](#)²³ ⌚ April 2, 2024 2:56 pm

Thank you!

👍 0 ➡ Reply

**Lux**

⌚ March 27, 2024 5:58 am

I have a problem where if I enter a number it outputs whatever number I put in, plus an additional extra digits.

So if I put in 5, I end up getting 52619 for some reason. Any help?

👍 0 ➡ Reply

**laughingwater**Reply to [Lux](#)²⁴ ⌚ March 27, 2024 8:42 am

Provide code, maybe because you might be using `'/n'` instead of `'\n'`

👍 0 ➡ Reply

**Nicolas**Reply to [Lux](#)²⁴ ⌚ March 27, 2024 7:56 am

Show us your code if you can. It will be easier to understand your problem

👍 0 ➡ Reply

**Lucas**

⌚ March 20, 2024 1:45 pm

How do I create multiple files with multiple main functions, is it possible or will I need to create a new project for every snippet?

👍 0 ➡ Reply

**Alex** AuthorReply to [Lucas](#)²⁵ ⌚ March 21, 2024 2:02 pm

A project can only have a single main() function.

It's probably going to be burdensome to create a new project for every example. I recommend having a single project where you can compile examples, overwriting the existing source code each time. If you decide a program is worth saving, create a new project.

That way you only create projects for the things you really want to keep.

👍 0 ➡ Reply

**Daur Daur Toogis**

⌚ March 15, 2024 11:43 am

Hi, might be a stupid question, but where do I provide my input when the program asks "Enter a number" ?

I see "Enter a number" in the debug console. If i type it there it does not work, and if i type it in the terminal it says that is not a command.

👍 0 ➡ Reply

**Alex** AuthorReply to [Daur Daur Toogis](#)²⁶ ⌚ March 17, 2024 10:05 pm

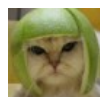
What IDE are you using?

👍 1 ➡ Reply

**Daur Daur Toogis**Reply to [Alex](#)²⁷ ⌚ March 19, 2024 3:29 pm

I am using Visual Studio Code on Mac. Clang++ is installed

👍 0 ➡ Reply

**Alex** AuthorReply to [Daur Daur Toogis](#)²⁸ ⌚ March 20, 2024 11:23 am

Try installing the Code Runner extension.

👍 0 ➡ Reply

**Daur Daur Toogis**Reply to [Alex](#)²⁹ ⌚ March 20, 2024 11:29 am

Will do. Much Appreciated!

👍 0

➡ Reply



laughingwater

🗨 Reply to [Daur Daur Toogis](#)³⁰ 🕒 March 27, 2024 8:44 am

Make sure to go into settings (CTRL + ,) and searching "Code Runner Run In Terminal" and checking the box, this use the actual terminal instead of the Output section of vscode

👍 0

➡ Reply



Tatsuya Yuki

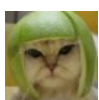
🕒 March 13, 2024 7:41 am

is there anyway to simplify or optimise this?

```
1  #include <iostream>
2
3  int main()
4  {
5      int a{}, b{};
6
7      // creates variables 'a' and 'b'
8
9      std::cout << "Enter a number: ";
10     std::cin >> a;
11
12     // user adds value to variable 'a'
13
14     std::cout << "Enter a second number ";
15     std::cin >> b;
16
17     // user adds value to variable 'b'
18
19     std::cout << "Both numbers added together is: " << a + b;
20
21     // adds together variable 'a' and 'b' and outputs it
22 }
```

👍 0

➡ Reply



Alex Author

🗨 Reply to [Tatsuya Yuki](#)³¹ 🕒 March 13, 2024 10:42 pm

Not really. I'd even move the definition of `b` to line 13.

👍 2

➡ Reply

**rakesh**Reply to [Alex](#)³² ⌚ March 21, 2024 9:30 pm

why is that?

👍 0

➡ Reply

**Alex** AuthorReply to [rakesh](#)³³ ⌚ March 22, 2024 11:36 am

Best to define variables as close to their first use as possible.

👍 0

➡ Reply

**Hakuriyuo**

⌚ February 26, 2024 8:28 am

procrastinated for 2 months finally reach this chapter today and i'd say i would keep going tomorrow.. its very interesting :p my c++ career path is finally opened

👍 0

➡ Reply

**Asmo**

⌚ February 16, 2024 9:38 pm

Can you please review this code and tell me if is there anything can i do it to simplify ?

```
1 | #include <iostream>
2 |
3 | int main()
4 | {
5 |     int a{};
6 |     int b{};
7 |     std::cout << " Input a=" ;
8 |     std::cin >> a ;
9 |     std::cout << "Input b=";
10 |    std::cin >> b;
11 |    int x = a+b;
12 |    std::cout << "The sum of a and b is :." << x << '\n';
13 |    return 0;
14 | }
```

✎ Last edited 1 month ago by Asmo

👍 0

➡ Reply

**EvilDeeds**Reply to [Asmo](#)³⁴ ⌚ February 28, 2024 6:18 am

just remove x and sum a + b in one line

```
1 | #include <iostream>
2 |
3 | int main()
4 | {
5 |     int a{};
6 |     int b{};
7 |     std::cout << " Input a=" ;
8 |     std::cin >> a ;
9 |     std::cout << "Input b=";
10 |    std::cin >> b;
11 |    std::cout << "The sum of a and b is :" << a + b << '\n';
12 |    return 0;
13 | }
```

👍 0

➡ Reply



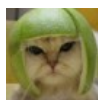
Likss

🗨 Reply to [EvilDeeds](#)³⁵ 🕒 April 6, 2024 12:10 pm

I just want to know where the compiler stores the value of x+y since no extra variable has been declared . Kindly Answer

👍 0

➡ Reply



Alex Author

🗨 Reply to [Likss](#)³⁶ 🕒 April 8, 2024 8:38 am

In a temporary (unnamed) object.

👍 0

➡ Reply



Likss

🗨 Reply to [Alex](#)³⁷ 🕒 April 9, 2024 12:58 am

Ok . Thanks

👍 0

➡ Reply



Noob

🗨 Reply to [Asmo](#)³⁴ 🕒 February 19, 2024 4:33 pm

I think you are set.

👍 0

➡ Reply



IceFloe

🕒 February 15, 2024 3:07 am

the lesson was really interesting, I would like more information about buffering, if possible, can you give a link to useful materials? And how does caching differ from buffering?

And also, I hope that they will teach us how to enter other characters from the keyboard, probably the data entered from the keyboard should depend on the type of variable that we have defined, but that's why if I entered the character `h`, it turns out zero, it's also interesting.

👍 1 ➡ Reply



Alex Author

🔗 Reply to [IceFloe](#)³⁸ ⌚ February 17, 2024 11:45 am

Wikipedia is always a good place to start for general concepts.

A buffer is a temporary storage location for information that is waiting to be processed. It's typically structured as a queue, and after the information is processed it is removed.

Having a buffer allows reading and writing data to the buffer to be desynchronized. Data is typically added to the buffer manually.

A cache is a temporary storage location for fast retrieval of that information, with the goal of improving performance. Computer architectures contain caches to reduce the number of memory requests, for example. Caches are often filled automatically.

We talk about inputting other types (and error handling) in future lessons.

👍 1 ➡ Reply



Bishoy S

⌚ February 11, 2024 10:12 am

In Q a) why the answer is zero not 104 (as ASCII value) ??

👍 0 ➡ Reply



Alex Author

🔗 Reply to [Bishoy S](#)³⁹ ⌚ February 11, 2024 11:50 pm

Because this would introduce ambiguity when entering a single value like `5`. Should this be interpreted as `5` (value 5) or `'5'` (value 53)?

If you want to extract a char based on it's ascii value, extract to an object of type `char`, not `int`.

👍 2 ➡ Reply



Vishal Balaji

⌚ February 10, 2024 10:38 pm

For some reason, declaring a variable as `int x{};` is failing on my compiler. If I just change it to `int x;` everything runs fine.

I tried a bunch of solutions found online but none of it is helping. Does anyone know what is causing this?

I'm using VS Code on Macbook

👍 0 ➡ Reply



Vishal Balaji

🔗 Reply to [Vishal Balaji](#)⁴⁰ ⌚ February 10, 2024 10:51 pm

Ok running as `clang++ -std=c++17 -g helloworld.cpp -o helloworld` instead of `clang++ helloworld.cpp -o helloworld` makes it run.

But I'm not sure why this is the case. Both `-g` and `-std=c++17` are part of my `tasks.json`

👍 1 ➡ Reply



Vishal Balaji

🔗 Reply to [Vishal Balaji](#)⁴⁰ ⌚ February 10, 2024 10:38 pm

It generates this error

```
1 | helloworld.cpp:24:10: error: expected ';' at end of declaration
2 |     int x{};           // define variable x to hold user input (and
3 |     value-initialize it)
4 |         ^
5 |         ;
    | 1 error generated.
```

👍 0 ➡ Reply



Asmo

🔗 Reply to [Vishal Balaji](#)⁴¹ ⌚ February 16, 2024 8:45 pm

you missed the semi column

👍 1 ➡ Reply



Kamil G.

⌚ February 7, 2024 12:09 pm

Fun discovery, if you have `int` and `double` one after another in `std::cin` and in your input you write number with fraction e.g. `5.6` this action divides this number in two variables of `int = 5` and `double = 0.6`

```
1 | #include <iostream>
2 |
3 | int main(){
4 |     int a {};
5 |     double b {};
6 |
7 |     std::cin >> a >> b;
8 |
9 |     std::cout << "a=" << a << " b=" << b;
10 |     return 0;
11 | }
```



1



Reply

**Alex** AuthorReply to [Kamil G.](#)⁴² February 7, 2024 12:27 pm

Yup, but I don't think it works if you type in a decimal number without a leading 0 (e.g. `.5`).



0



Reply

**Corbin**

February 4, 2024 12:23 pm



```
1 | #include <iostream>
2 |
3 | int main() {
4 |
5 |     std::cout << "Any Number = ";
6 |     int a{10};
7 |     int b{10};
8 |
9 |     std::cin >> a;
10 |
11 |     std::cout << "Another Number = ";
12 |
13 |     std::cin >> b;
14 |
15 |     std::cout << "Your First Number Is... " << a << '\n';
16 |     std::cout << "Your Second Number Was... " << b << '\n';
17 |
18 |     std::cout << "Total Result = " << a << b;
19 |
20 |
21 |     return 0;
22 | }
```

Proud of this :strongemoji::strongemoji::strongemoji:

Last edited 2 months ago by Corbin



3



Reply

**Howard**

🕒 February 1, 2024 3:27 am

Hey Alex, my lines were not working and told me that I should add ";" at the end of declaration `int x{}` though I've added `;`. However, when I typed `x=0` it worked. I think the problem with it was my compiler version, but I've already updated my compiler to the latest version of c++ 11. Could you please help me solve this issue? Thank you very much!

👍 0

➡ Reply

**Alex** Author🗨 Reply to Howard⁴³ 🕒 February 2, 2024 9:57 am

Have you selected the correct language standard? Your compiler may still be defaulting to an older version of C++. See <https://www.learncpp.com/cpp-tutorial/configuring-your-compiler-choosing-a-language-standard/>

👍 0

➡ Reply

**zGrvs**🗨 Reply to Alex⁴⁴ 🕒 February 9, 2024 10:24 am

I am having the same issue and also on macOS, C++20 standard. `int x;` compiles fine however. Can we get away with initializing in this way?

EDIT: This issue has opened up more questions. I thought I could compile from the command line with `clang++ file.cpp`. This is the command that was not allowing `int x{}; ...` But when I compile with the run and debug menu in my IDE, it compiles perfectly fine. I'm slightly confused. Is it because command line is using a system-based compiler, while my IDE uses a different one? For now, I will continue using the Run button in my IDE, but I would eventually like to be able to use `clang++` from terminal just from an ease of use perspective

📝 Last edited 2 months ago by zGrvs

👍 0

➡ Reply

**Alex** Author🗨 Reply to zGrvs⁴⁵ 🕒 February 9, 2024 2:00 pm

Initial question: No.

Have you tried compiling from the command line `clang++ -std=c++20 file.cpp`?

📝 Last edited 2 months ago by Alex

👍 1

➡ Reply

**zGrvs**Reply to [Alex](#)⁴⁶ February 9, 2024 2:12 pm

I had intended on editing once again after work but you beat me to it! That was precisely the problem; macOS clang defaults to gnu++98, so arg `-std=c++20` did allow the program to compile from the terminal. There doesn't seem to be a way to change this default. I find it very strange it defaults to such an old standard. Granted, I'm very new to all this so perhaps there is some reason for that beyond my grasp.

Thanks for the response! These lessons are great and I appreciate your ongoing support.

👍 0 ➡ Reply

**Howard**Reply to [Alex](#)⁴⁴ February 7, 2024 12:48 am

Yeah I tried. I searched it online, and I found that many other mac users have the similar issue. I guess it's the system.

👍 0 ➡ Reply

**Kenny**

February 1, 2024 1:14 am

for the: "std::cout" or "std::cin" why not use the "using namespace std" library to simplify it and not need to put "std::" everytime ?

👍 0 ➡ Reply

**Alex** AuthorReply to [Kenny](#)⁴⁷ February 2, 2024 9:51 am

<https://www.learncpp.com/cpp-tutorial/cpp-faq/#usingnamespace>

👍 3 ➡ Reply

**Slaven**

January 27, 2024 1:00 pm

Good explanation, and good job, the best site for learning , congrats and continue...

👍 1 ➡ Reply

**UzumakiMadara**

🕒 January 27, 2024 9:36 am

```
1 | std::cout<<"first print statement \n";
2 | std::cout<<"second print statement ";
```

Since \n doesn't necessarily flush output into the console it is possible that even during the execution of second statement the first text is still not printed to the console, right? So, doesn't that mean we can't completely rely on logs for debugging purpose in this cases?

👍 0

➡ Reply

**Alex** Author👤 Reply to [UzumakiMadara](#)⁴⁸ 🕒 January 31, 2024 5:57 pm

Output is queued in order of execution.

Execution could happen in one of two ways:

- Line 1 queues its output. Then a flush happens, printing "first". Line 2 queues its output. Then a flush happens, printing "second".
- Line 1 queues its output. No flush happens. Line 2 queues its output. Then a flush happens, printing "first" and then "second".

Either way, "first" will always be printed before "second".

However, if for some reason the program crashes, any queued output is lost. So it's possible that "first" could be output and not "second". It's also possible neither get printed.

If you want to ensure output is flushed immediately for debugging purposes, you can use `std::unitbuf`. We discuss this in lesson <https://www.learncpp.com/cpp-tutorial/using-an-integrated-debugger-stepping/>

👍 0

➡ Reply

**Victor**

🕒 January 21, 2024 6:25 am

My program works fine when directly executing from VS Studio (ctrl+f5). But the console window blinks and soon closes if I open it in the file explorer after I enter a number.

```
#include <iostream>
```

```
int main()
```

```
{
```

```
std::cout << "Enter your age, please.\n";
```

```
int x;
```

```
std::cin >> x;
```

```
std::cout << "You are " << x << " years old";
```

```
return 0;  
}
```


 0 Reply**Alex** Author Reply to [Victor](#)⁴⁹  January 23, 2024 9:13 am

This is just how Windows behaves. If you want it to not auto-close, you have several options:

1. Don't run it from file explorer, open a console window and run it from there.
2. Just run it from Visual Studio
3. Add some code at the bottom of main that waits for user input

 0 Reply**Victor** Reply to [Alex](#)⁵⁰  January 23, 2024 9:36 pm

Okay, thanks for answering:)

 0 Reply**DDDD** January 17, 2024 5:55 pm

I apologize if my question doesn't relate to this topic:

```
1 | #include <iostream>  
2 |  
3 | int main()  
4 | {  
5 |     std::cout << true + "Hello";           // print "ello" to console  
6 |     std::cout << true + true + "Hello";    // print "llo" to console  
7 |  
8 |     return 0;  
9 | }
```

I played with the above snippet of code and found it weird. I googled it, but no results came up, so I'm posting here. Could someone please explain this? Thank you.

 0 Reply**Alex** Reply to [DDDD](#)⁵¹  January 18, 2024 11:51 am

In C++, the true boolean value is internally represented as 1. When you use the + operator with a boolean and a string literal, the boolean is implicitly converted to its numeric value.

So when you do `std::cout << true + "Hello";` true is treated as 1, and when added to the string literal "Hello," it effectively shifts the output to start from the second character, resulting in "ello" being printed to the console.

👍 4 ➡ Reply



DDDD

🗨 Reply to [Alex](#) ⁵² ⌚ January 20, 2024 4:56 am

Okay, thank you Alex.

👍 0 ➡ Reply



Shiloh

⌚ January 15, 2024 3:11 am

Thanks for all the tutorials Alex, love this website!

👍 1 ➡ Reply



NoobCoder

⌚ January 12, 2024 1:15 pm

> "Prefer '\n' over `std::endl` when outputting text to the console."
Why Bjarne Stroustrup made `std::endl`?

👍 1 ➡ Reply



Alex Author

🗨 Reply to [NoobCoder](#) ⌚ January 13, 2024 8:40 pm

I presume as a convenient and more efficient way to do a newline and a flush together. Perhaps when the language was designed this was more important for some reason.

👍 0 ➡ Reply



Jordan

⌚ January 11, 2024 3:11 pm

it is always necessary to write "std" before cout and endl?

👍 0 ➡ Reply

**Alex** AuthorReply to [Jordan](#) January 11, 2024 10:47 pm

Necessary? No. There are ways to avoid having to do so.

But you really should.

1 Reply

**Gripps**

January 7, 2024 5:36 pm

The summary section here is incredibly helpful, thank you for including it.

0 Reply

**Machitot**

December 31, 2023 6:44 pm

in my code why i enter 5.5 for x and 6.5 for y, when I print value result is 5 and 0, it's not 5 for x and 6 for y

```
#include<iostream>
```

```
int main() {
```

```
std::cout << "Enter your value seperate by space: ";
```

```
int x{};
```

```
int y{};
```

```
std::cin >> x >> y;
```

```
std::cout << "value x,y is " << x << ',' << y << "\n";
```

```
return 0;
```

```
}
```

0 Reply

**voidboy**Reply to [Machitot](#) January 2, 2024 8:02 am

Because you asked to read an int but gave a double, std::cin failed to read past 5, thus, .5 6.5 was discarded and y stayed at 0.

Please note that if you want to provide both values at once(separated by space), you have to parse those, std::cin won't do it for you.

3 Reply

**Andy**

🕒 December 29, 2023 10:50 pm

```
#include <iostream>
```

```
int main()
```

```
{
```

```
std::cout << "Enter a number\n";
```

```
std::cout << " \n"; // * These are just to add a space to the text when shown.
```

```
int x{};
```

```
std::cin >> x;
```

```
std::cout << " \n"; // *
```

```
std::cout << "You have entered " << x << '\n';
```

```
std::cout << " \n"; // *
```

```
std::cout << "Enter two numbers with spaces\n";
```

```
std::cout << " \n"; // *
```

```
int j{};
```

```
int k{};
```

```
std::cin >> j >> k;
```

```
std::cout << " \n"; // *
```

```
std::cout << "You have entered " << j << " " << k;
```

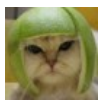
```
return 0;
```

```
}
```

Is this the best way to add spaces in-between lines of text? I imagine there is a more simplified way to do this.

👍 0

➡ Reply

**Alex**

Author

🗨 Reply to [Andy](#) 🕒 January 2, 2024 11:56 am

Why would you want to add a space to an otherwise blank line?

👍 0

➡ Reply

**xhinmu**

🕒 December 19, 2023 2:59 pm

```
1  std::cout << "Enter a number: ";
2
3  int x{};
4  std::cin >> x;
5
6  std::cout << "You entered " << x << '\n';
7
8  std::cout << "Enter two numbers separated by a space: ";
9
10 int a{};
11 int b{};
12 std::cin >> a >> b;
13
14 std::cout << "You entered " << a << " and " << b << '\n';
```

This is my program where I was testing out the quiz questions. When I type a letter, fractional component, or an integer that surpasses the maximum int for x, I noticed that it doesn't prompt me to type numbers for a and b and immediately says I typed 0 and 0. What causes this behavior? Is it something like buffer overflow?

👍 1 ➡ Reply



rakshxt

🗨 Reply to [xhinmu](#) 🕒 December 22, 2023 2:45 am

i would prefer using the following rather than the lines 10 & 11

```
1 | int a{}, b{};
```

also, try to define/initialize the variables at the start of main function

👍 0 ➡ Reply



Alex Author

🗨 Reply to [xhinmu](#) 🕒 December 20, 2023 1:40 am

We cover why this happens and what to do about it in future lesson
<https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/>

👍 0 ➡ Reply

**operation n313**

🕒 December 14, 2023 5:32 am

When I enter anything other than integer in the last program, like 'h' or a fractional component, it does not output anything and just closes the console immediately. Otherwise it works fine. Why is that?

👍 0

➡ Reply

**AbeerOrTwo**

🕒 December 13, 2023 8:24 pm

Started typing out the coding examples and has been helping a lot

👍 0

➡ Reply

**issame**

🕒 December 9, 2023 1:45 pm

I have a dumb question, what if I want `\n` to be a part of my console output. Would creating a string be the solution?

👍 0

➡ Reply

**issame**🗨 Reply to [issame](#) 🕒 December 19, 2023 7:54 am

Thank you guys so much! :)

👍 0

➡ Reply

**Gootlier**🗨 Reply to [issame](#) 🕒 December 13, 2023 5:35 am

good question, use `"\n"`. Here the first let the computer know to print the following `"\n"` as it should be.

👍 1

➡ Reply

**dfjhgfjvgdsjfgsdfjfgi**🗨 Reply to [issame](#) 🕒 December 10, 2023 2:23 am

do `\\n` instead of `\n`

👍 1

➡ Reply

Links

1. <https://www.learncpp.com/author/Alex/>
2. <https://www.learncpp.com/cpp-tutorial/naming-collisions-and-an-introduction-to-namespaces/>
3. <https://www.learncpp.com/cpp-tutorial/chars/>
4. <https://www.learncpp.com/cpp-tutorial/a-few-common-cpp-problems/>
5. <https://www.learncpp.com/cpp-tutorial/stdcin-and-handling-invalid-input/>
6. <https://pdcurses.org/>
7. <https://github.com/ArthurSonzogni/FTXUI>
8. <https://github.com/jupyter-xeus/cpp-terminal>
9. <https://github.com/dankamongmen/notcurses>
10. <https://www.learncpp.com/cpp-tutorial/introduction-to-literals-and-operators/>
11. [javascript:void\(0\)](javascript:void(0))
12. <https://www.learncpp.com/cpp-tutorial/signed-integers/>
13. <https://www.learncpp.com/cpp-tutorial/uninitialized-variables-and-undefined-behavior/>
14. <https://www.learncpp.com/>
15. <https://www.learncpp.com/cpp-tutorial/variable-assignment-and-initialization/>
16. <https://www.learncpp.com/introduction-to-iostream-cout-cin-and-endl/>
17. <https://www.learncpp.com/site-news/site-migration/>
18. <https://www.learncpp.com/cpp-tutorial/introduction-to-function-parameters-and-arguments/>
19. <https://gravatar.com/>
20. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595581>
21. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595519>
22. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595235>
23. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595266>
24. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595157>
25. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594899>
26. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594730>
27. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594801>
28. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594869>

29. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594891>
30. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594893>
31. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594628>
32. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594666>
33. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594961>
34. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593739>
35. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-594137>
36. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595489>
37. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-595534>
38. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593649>
39. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593535>
40. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593514>
41. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593515>
42. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593373>
43. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593112>
44. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593159>
45. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593464>
46. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593472>
47. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-593108>
48. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-592955>
49. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-592658>
50. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-592732>
51. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-592481>

52. <https://www.learncpp.com/cpp-tutorial/introduction-to-iostream-cout-cin-and-endl/#comment-592529>